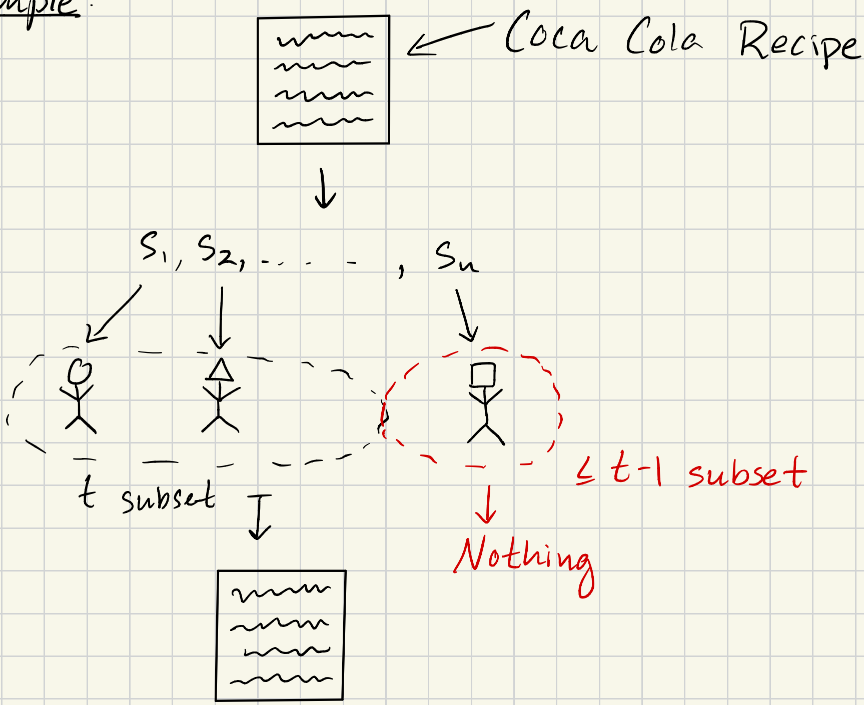
Suppose I have a secret that I want to share across n parties s.t. any t subset can recover my secret, but any $\leq t-1$ subset learn nothing about my secret

Example:



Definition: A (t,n)-secret sharing scheme over a message space $\mathcal{M}$ and share space $\mathcal{S}$ is a tuple of efficient algs:

$$\text{Share}: \mathcal{M} \to \mathcal{S}^n$$
$$\text{Reconstruct}: \mathcal{S}^t \to \mathcal{M}$$

with the following properties:

Made with Goodnotes

<u>Correctness</u>: any t shares can be used to reconstruct m.

$$\forall m \in \mathcal{M},\ (s_1, \dots s_n) \leftarrow \text{Share}(m)$$
$$\forall S \subseteq \{s_1, \dots, s_n\} \text{ where } |S| = t$$
$$\text{Reconstruct}(S) = m$$

<u>Security</u>: need at least t shares to learn anything about m.

$$\forall m_0, m_1 \in \mathcal{M},\ \forall I \subseteq \{1, \dots, n\} \text{ where } |I| < t:$$

Denote $(s_1, \dots, s_n) \leftarrow \text{Share}(m_0)$
$(s_1', \dots, s_n') \leftarrow \text{Share}(m_1)$

$$\{s_i \mid i \in I\} \approx \{s_i' \mid i \in I\}$$

↖ today, these distributions will be identical

<u>Construction of (n,n)-secret sharing</u>

For message space $\mathcal{M} = \mathbb{F}$ and $S = \mathbb{F}$ ← can just be abelian group

<u>Share(m)</u>: Sample $r_1, \dots, r_{n-1} \xleftarrow{\$} \mathbb{F}$
Define $r_n := m - \sum_{i=1}^{n-1} r_i$
Output $(r_1, \dots r_n)$

<u>Reconstruct$(r_1, \dots, r_n)$</u>: Output $m' := \sum_{i=1}^{n} r_i$

<u>Correctness</u>: $\sum_{i=1}^{n} r_i = \sum_{i=1}^{n-1} r_i + \left(m - \sum_{i=1}^{n-1} r_i\right) = m$

<u>Security</u>: $\forall m_0, m_1 \in \mathcal{M}$, the $(n-1)$ share distributions are actually identical

Made with Goodnotes

Shamir Secret Sharing: (t,n)-secret sharing scheme

For $\mathcal{M} = \mathbb{F}$, $S = \mathbb{F}$ s.t. $|\mathbb{F}| > n$

Intuition: a polynomial of degree $t-1$ can be uniquely determined by t points. For example, a line is defined by 2 points, and a parabola is defined by 3.

Linear Algebra Viewpoint: Given a point $x \in \mathbb{F}$ and a poly $f(X) = c_0 + c_1 X + c_2 X^2 + \dots + c_{t-1} X^{t-1}$
We can view the evaluation $f(x)$ as an inner product:

$$f(x) = \langle (1, x, x^2, \dots, x^{t-1}), (c_0, c_1, \dots, c_{t-1}) \rangle$$

Thus, given t distinct points, we can describe a linear system:

$$\underbrace{\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{t-1} \\ 1 & x_1 & x_1^2 & \cdots & x_1^{t-1} \\ & & \ddots & & \\ 1 & x_{t-1} & x_{t-1}^2 & & x_{t-1}^{t-1} \end{bmatrix}}_{\text{Vandermonde Matrix}} \underbrace{\begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{t-1} \end{bmatrix}}_{\text{coeffs}} = \underbrace{\begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_{t-1}) \end{bmatrix}}_{\text{evals}}$$

Interpolating $f(x)$ is equivalent to solving the linear system above for the coeffs. For distinct $x_0, \dots, x_{t-1}$, the Vandermonde matrix is invertible; thus, interpolating requires a matrix mul $V^{-1} \cdot \text{evals}^T$

Proof Sketch of Invertibility

If cols linearly dependent, then $\exists\, c_0, \dots, c_{t-1}$ s.t.:

$$c_0 \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} + c_1 \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{t-1} \end{bmatrix} + \dots + c_{t-1} \begin{bmatrix} x_0^{t-1} \\ x_1^{t-1} \\ \vdots \\ x_{t-1}^{t-1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Made with Goodnotes

If this were true, then $\overbrace{x_0, \ldots x_{t-1}}^{t}$ are roots of a poly of deg $\leq t-1$ w/ coeffs $c_0, \ldots c_{t-1}$. BUT a poly of deg $t-1$ has at most $t-1$ roots. Thus, $c_0, \ldots c_{t-1} = 0$

Construction

Share(m): Sample random coeffs $c_1, \ldots, c_{t-1} \xleftarrow{\$} \mathbb{F}$
Define $f(X) := m + \sum_{i<t} c_i X_i$
Output n points on f:
$(s_i := (i, f(i)) \ \forall i \in [1, n])$

Reconstruct $((x_i, y_i), \forall i \in [t])$
- Interpolate the unique poly f of deg $t-1$ defined by those t points
- Output $f(0)$

Correctness: Follows from the uniqueness of interpolation (t points define a poly deg $\leq t-1$)

Security: Consider an arbitrary message $m \leftarrow \mathbb{F}$ and $I \subseteq [1, n]$ s.t. $|I| = t-1$. Define $\{x_i \mid \forall i \in [1, t-1]\} := I$. Consider arbitrary $y_1, \ldots, y_{t-1} \in \mathbb{F}$. What's the probability the shares for this subset are $(x_1, y_1), \ldots, (x_{t-1}, y_{t-1})$?

$$\Pr\left[V(0, x_1, \ldots, x_{t-1}) \begin{bmatrix} m \\ c_1 \\ \vdots \\ c_t \end{bmatrix} = \begin{bmatrix} m \\ y_1 \\ \vdots \\ y_{t-1} \end{bmatrix} \right]$$

$$= \Pr\left[\begin{bmatrix} m \\ c_1 \\ \vdots \\ c_{t-1} \end{bmatrix} = \underbrace{V^{-1}(\ldots) \begin{bmatrix} m \\ y_1 \\ \vdots \\ y_{t-1} \end{bmatrix}}_{\text{a fixed } \hat{u} \in \mathbb{F}^n} \right] = \frac{1}{|\mathbb{F}|^{t-1}}$$

independent of m!

Made with Goodnotes

Another way to interpret: for any choice of $(t-1)$ shares and any message m, there exists a unique poly f of deg $t-1$ s.t.

$$\forall i \in [1, t-1],\ f(x_i) = y_i \text{ and } f(0) = m$$

Thus, any $(t-1)$ shares can be consistent with the sharing of any message m.

Now, we will describe a protocol for 2-PC (two-party MPC) for functions, expressible by Arithmetic Circuits

We will show an elegant construction from secret sharing that targets semihonest security, where we restrict corrupt parties to follow the protocol specification exactly but can try to extract info about the honest parties' input.

* there is an elegant way to make the protocol maliciously secure by using "info-theoretic MACs"

2-Party Computation for Arithmetic Circuits

	Alice (x_A)		Bob (x_B)
1. Share Inputs	$[x_A]_0, [x_A]_1 \leftarrow Share(x_A)$		$[x_B]_0, [x_B]_1 \leftarrow Share(x_B)$
		$\xrightarrow{[x_A]_1}$	
		$\xleftarrow{[x_B]_0}$	
2. Compute on secret shares	⋮		⋮

Made with Goodnotes

$[\text{Output}]_0$ $\qquad$ $[\text{Output}]_1$

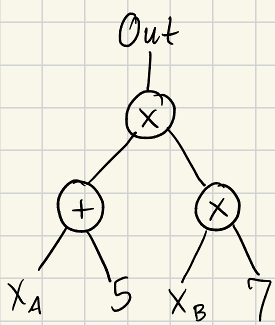
3. Open final output shares $\qquad$ $\xrightarrow{[\text{Output}]_0}$ $\qquad$ $\xleftarrow{[\text{Output}]_1}$

4. Reconstruct $\qquad$ $[\text{Output}]_0 + [\text{Output}]_1 = \text{Output}$

Protocol

Idea: derive shares to intermediate wires incrementally

1. Both parties secret share their input elements with each other
2. For each addition gate in the circuit with inputs $[x]$, $[y]$, the parties jointly derive shares to $[x+y]$ (the output share)
3. For each multiplication gate w/ inputs $[x]$, $[y]$, parties jointly derive shares to $[x \cdot y]$
4. Once share of circuit outputs is derived, each party sends their share of the output



Alice		Bob
$[x_A]_0, [x_A]_1 \leftarrow \text{Share}(x_A)$		$[x_B]_0, [x_B]_1 \leftarrow \text{Share}(x_B)$
	$\xrightarrow{[x_A]_1}$	
$[x_A]_0, [x_B]_0$	$\xleftarrow{[x_B]_0}$	$[x_A]_1, [x_B]_1$
$\downarrow$		$\downarrow$
$[x_A+5]_0, [7x_B]_0$	$\rightleftarrows$	$[x_A+5]_1, [7x_B]_1$
$\downarrow$		$\downarrow$
$[(x_A+5)(7x_B)]_0$		$[(x_A+5)(7x_B)]_1$
$\parallel$		$\parallel$
$[\text{Out}]_0$	$\xrightarrow{[\text{Out}]_0}$	$[\text{Out}]_1$

Made with Goodnotes

$$[Out]_1$$

$$\leftarrow$$

$$[Out]_0 + [Out]_1 = Out$$

Computation on Secret Shares

Here we assume the (2,2) scheme above $[x]_0 = r \xleftarrow{\$} \mathbb{F}$, $[x]_1 = x - r$

Adding Shares: $[x_A]_0 + [x_B]_0 = [x_A + x_B]_0$

$$[x_A]_1 + [x_B]_1 = [x_A + x_B]_1$$

$$[x_A + x_B]_0 + [x_A + x_B]_1 = [x_A]_0 + [x_B]_0 + [x_A]_1 + [x_B]_1 = x_A + x_B$$

Adding a constant c: $c/2 + [x_B] = [c + x_B]$

Multiplying by a constant c: $c[x_B] = [cx_B]$

Multiplying Shares: will require setup + interaction

Beaver's Trick 91

Suppose parties already have secret shares of a random product: $\underbrace{[a], [b], [c]}_{\text{Beaver triple}}$ where $a, b \xleftarrow{\$} \mathbb{F}$ and $c = ab$

To multiply shares $[x]$ and $[y]$:

from adding shares

1. Locally compute shares to $[x-a]$ and $[y-b]$
2. Send shares to jointly reconstruct $\varepsilon = x - a$ and $\delta = y - b$; note that ε and δ are both one time pad encryptions of x and y
3. Locally compute shares $[z] = [c] + \delta[x] + \varepsilon[y] - \frac{\varepsilon\delta}{2}$

Correctness

$$[z]_0 + [z]_1 = [c]_0 + \delta[x]_0 + \varepsilon[y]_0 - \frac{\varepsilon\delta}{2}$$
$$+$$
$$[c]_1 + \delta[x]_1 + \varepsilon[y]_1 - \frac{\varepsilon\delta}{2}$$

Made with Goodnotes

$$
\begin{aligned}
&= c + \delta x + \varepsilon y - \varepsilon\delta \\
&= ab + (y-b)x + (x-a)y - \varepsilon\delta \\
&= ab + xy - bx + xy - ay - (x-a)(y-b) \\
&= ab + xy - bx + xy - ay - (xy - bx - ay + ab) \\
&= xy
\end{aligned}
$$

<u>Security</u>: Information-theoretic!

How do parties obtain Beaver triples?

* Requires public-key cryptography or a trusted dealer
  - ↳ Oblivious Transfer
  - ↳ Garbled Circuits
  - ↳ Somewhat homomorphic encryption

Need to generate one Beaver triple per multiplication gate (cannot reuse triples o/w break OTP)

2-PC in the Preprocessing Model

<u>Preprocessing / offline stage</u>: parties generate M Beaver triples where M is an upper bound on multiplication gates. Note that this process is expensive, but independent of the future circuit / party input

- save these Beaver triples for use in the future; maybe waiting for data to be available or a computation to be agreed on

Made with Goodnotes

Online Stage: no expensive PK operations but requires communication linear in the # of mult. gates / inputs

- parties secret share inputs
- parties perform the 2-PC protocol using pregenerated triples (as long as # mult gates < M)
- parties reconstruct output

Made with Goodnotes